Overview of ME964 Final Projects
April 14, 2011

“The first 90 percent of the code accounts for the first 90 percent of the development time. The remaining 10 percent of the code accounts for the other 90 percent of the development time.”
—Tom Cargill
Before We Get Started…

- **Last Time**
  - Parallel programming patterns
  - One slide summary of ME964

- **Today**
  - Quickly go through your Final Project proposal presentations

- **Other issues:**
  - Assignment 8 (last ME964 assignment) due today at 11:59 PM
  - Midterm exam on April 19
    - Review session the evening before
    - Syllabus called for closed books, it’s ok to bring whatever source of information you want with you
  - From now on only guest lectures and such, time to concentrate on your projects
  - **IMPORTANT:** Final Project Proposal PDF doc due on Tu, at 11:59 PM
    - You submitted a proposal and/or provided a 3 slide presentation on the topic of your choice
    - If you don’t hear back from me it means that I’m ok with the topic you proposed and you should get going on your Final Project
    - I will use Doodle to allow you to enter the time/day when you choose to present your Final Project work
Discretized Poisson Sparse Matrix Solver with Conjugate Gradient Method

Spring 2011

Lakshman Anumolu

Department of Mechanical Engineering
University of Wisconsin, Madison
Motivation & Illustration

- Fluid flow simulations require the solution of Poisson equation.
- Solving Poisson equation is computationally expensive for high fidelity simulations.
- Discretized Poisson matrix contains at most 5 non-zero elements in each row.

\[
A = \begin{bmatrix}
4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\
-1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4
\end{bmatrix}
\]

(Example: For 5x5 nodes with known boundary conditions)\(^1\)

Goal

- **Midterm project:**
  - Implementation of Conjugate Gradient method for solving above system on GPU.

- **Final project:**
  - Extending above implementation by modifying the method to use symmetric nature of discretized matrix.
  - Comparing the performance with sparse matrix solver “SpeedIT_classic²”.

Isotropic Projection Gridding for MRI Applications

Mike Loecher

Problem: Grid 3D radial data onto a Cartesian grid so that an FFT can be performed to obtain our final image
Reasoning and prior work

• Gridding is the biggest time sink in the full image reconstruction
• Large data sets (~10,000,000 points per image x multiple timeframes)
• Potential to get a full reconstruction to a doctor while the patient is still in the scanner
• Some prior work by another author showed a speedup of around 100x, but in 2D and with a different trajectory
Outcomes

• Implement gridding for 3D isotropic projection data
• Obtain images with very little error compared to CPU gridding
• Optimize handling of variable sampling density (probably by separating into dense and sparse regions and handling differently)
Modeling the effects of applied stresses on granular materials with the discrete element method

Ian C. Olson

Current GPU DEM code:
- Normal Stiffness $K_n$
- Fixed boundaries
- Fixed particle size

New GPU DEM code:
- Add particle friction $F_f$
- Add capacity for variability in particle size
- Fixed, smooth side boundaries
- Smooth top / bottom boundaries apply stress to highest / lowest particles
- Output force distribution in system
- Automated variation in $\sigma$ when system reaches equilibrium
- Output changes in system height $h$ with corresponding applied stress
Final Project Problem Statement

- Develop GPU friendly code to compute a preconditioner used in the finite element analysis of thin structures.
GPU Friendly Preconditioners

- Why?
  - Many research projects, including my own, involve the solution of a linear system that is either impractical or impossible to solve directly. Using a preconditioner in iterative solvers can greatly reduce computation time and the number of iterations to solve the system but they are not easily implemented on a GPU.

- Preliminary Results
  - Dr. Suresh has already developed parts of the CUDA code and theory to implement a preconditioner on a GPU, as well as Abhirami and I have already implemented the calculation of the general metric over a triangular mesh and are validating and investigating the accuracy of our code.
GPU Friendly Preconditioners

- **Deliverables**
  - CUDA C code to complete the following tasks
    - Accurately calculate the general metric over the elements of a 3D triangular mesh
    - Calculate the general metric of a 3D structure for the volume inside of a bounding box
    - Calculate the stiffness matrix of a 3D structure using a general metric formulation
Final Project

- Brian J. Davis
- Medical Physics / BME
- Conebeam Backprojection Reconstruction for C-Arm CT and associated algorithms
- Need for real time cone-beam backprojections of projection images for four dimensional digital subtraction angiography (4D DSA)

Example of C-Arm CT - Siemens Artis Zeego shown
Rational

- Allow for 4D (3D time resolved volumes) to be viewed and rotated to any angle by the Radiologist.
- Allow selection of Region of Interest (ROI) to be selected showing perfusion, pulse, and Time of Arrival by the Radiologist.
Summary of Outcomes (Deliverables)

- Decrease backprojection reconstruction by the greatest degree possible. Current 512x512x397 reconstruction ~10 min on Tesla c1060. Matlab version was 2.5 days. Goal < 1 minute.
- Port other stages of recon to GPU currently in MATLAB with Mex interface. Log subtract, Parker Weights, Cosine Weighting, 4D-DSA, Time Interval Difference (TID), Color mapping, rendering (VTK), etc.
- Forward Projection (time allotting) – Reverse of back projection. Requires solving a sparse matrix of a linear system of equations.
Final Project Proposal
“Fast Normalized Cross Correlation”
ME 964
Kwang Won Choi
Abstract

- The correlation between two signals (cross correlation) is a typical approach to feature detection.
- The basic idea of this algorithm is finding the correlation coefficient between template feature (also called kernel) and input images.
The normalized cross-correlation between two signals of length $N$ is defined as:

$$r_{xy} = \frac{\sum_{n=1}^{N} \{x(n) - \bar{x}\}\{y(n) - \bar{y}\}}{\sqrt{\sum_{n=1}^{N} \{x(n) - \bar{x}\}^2 \sum_{n=1}^{N} \{y(n) - \bar{y}\}^2}}$$

The result is that $r_{xy}$ approaches 1 only when the region of image contained the template.
Usage in research field

- Elastography in the field of bio-research is a non-invasive technique in which stiffness or strain images of soft tissue are used to detect or distinguish tumours or tissues that have problems.
- We can detect and track a specific tissue out of whole tissue by using Cross Correlation
Fig 1. Normalized Cross Correlation used in the area of elastography to detect a specific tissue out of ultrasound image of whole tissue.
Optical Lens Optimization

- Kuya Takami
- Mechanical Engineering

- Optical fiber fused lens coupling optimization using CUDA programming for ray tracing.
Background

path length, $p = 80$ mm
$L_t = 5$ [mm]
$L_r = 16$ [mm]
By the end of the semester...

- Provide lens parameter optimizer program
- Based on user input (machining tolerance, parameter constraints) output optimal transmission lens design parameters.
Problem Statement

• Implement a direct parallel solver to solve very large sparse linear systems (millions of unknowns)

Reasons

• Solving sparse linear systems is needed in optimization, mechanics, fluidics, etc. The parallel direct solver remains an open problem, and there is so far no open source GPU implementation.

Preliminary results

• Investigated the SPIKE algorithm.
The SPIKE Algorithm

Goal: Solving $AX=b$, where $A$ is sparse
- Use some recording technique, transform $A$ to a banded matrix
- Partition $A$ into $p$ blocks (in this example, $p=3$)

$A = \begin{bmatrix}
A_1 \\
B_1 \\
C_1
A_2 \\
B_2 \\
C_2
A_3
\end{bmatrix}$

- Factorize $A$, $A=DS$, where $D=\text{diag}(A_1, \ldots, A_p)$

$S = \begin{bmatrix}
I & V_1 \\
W_2 & I & V_2 \\
W_3 & I
\end{bmatrix}$
Problem becomes solving DSX=b, can be divided into two steps:
(a) DG=b can be solved directly
(b) SX=G can be further reduced and solved either directly or iteratively depending on the problem type.

Expected Outcomes
- Implement SPIKE algorithm on GPU or other parallel scheme
- Optimize the code
Sparse Matrix Reordering

Spring 2011

Andrew Seidl

Department of Mechanical Engineering
University of Wisconsin, Madison
Sparse Matrix Reordering

- Goal: rearrange elements so that matrix becomes a banded matrix
  - Allows a partitioning of a large matrix as required by the SPIKE algorithm (see previous presentation)
  - This in turn enables the use of multiple MPI-connected nodes in solving the large sparse linear system

- ParMETIS
  - MPI-based library implementing multilevel nested dissection algorithm
  - Balanced elimination trees: allows for parallel direct factorization
Sparse Matrix Reordering

nz = 9240

nz = 9240
ME964 Final Project:
GPU Implementation of Absolute Nodal Coordinate Formulation (ANCF)

Dan Melanz – Mechanical Engineering Simulation-Based Engineering Lab
What is ANCF?

- Used to carry out the dynamics analysis of flexible bodies that undergo large rotation and large deformation

- Consistent with nonlinear theory of continuum mechanics, yet easy to implement

- Can be combined with the Discrete Element Method to handle friction and contact between beams
Applications of ANCF

- Hair simulations
- Polymer simulations
- Grass/Terrain modeling
- Material modeling
Parallelizing Exhaustive Search of Feasible Aperiodic Task Start

Rehan Ahmed
Realtime Tasks

- Tasks in which the correctness of a computed output is dependent not only on the logical correctness but also time at which the output is delivered.
- Realtime tasks have a notion of deadline.
- Depending on the type of system, missing a deadline can have catastrophic consequences.

Types Of Tasks
- **Periodic Tasks**: Repeat After every Time Period.
- **Aperiodic Tasks**: can come at arbitrary time instant.
Problem Definition

- In a given thermally constrained system, a static schedule for periodic tasks can be constructed such that their deadlines are met and thermal constraints for the system are met.
- Aperiodic tasks can be scheduled in idle times as long as the processor temperature does not go beyond a certain threshold level.
- Goal of the scheduler is to exhaustively search for the earliest start time for the aperiodic task.
- For this project, I intend to parallelize the exhaustive search phase of this algorithm.
  - Each thread can evaluate temperature impact at a different start time.
  - Earliest feasible start time is accepted
Deliverables

- Code for scheduling periodic/aperiodic tasks along with instructions (readme.txt)
  - Both serial and parallel versions
- Design Document:
  - Explanation of temperature estimation scheme.
  - Algorithm for making admission control decision of aperiodic tasks.
  - Performance analysis (serial vs. parallel)
Beam and plate structures are used extensively in structural engineering. Analysis of such structures involves the construction of a beam (or plate) stiffness matrix.
The parallelization achievable in GPUs enables the use of simpler algorithms at various stages involved in the analysis, and also provides significant speedup in the computations.

- Weighted volume computation has been implemented on the GPU and verified using various test structures and cases.
- Also part of the Midterm Project is to complete the calculation of the weighted volume of a model within a given bounding box.
SUMMARY OF DELIVERABLES

- For the final project, this framework is to be extended in order to calculate the beam stiffness matrix of any arbitrary structure, given a triangulated representation of it and 1-D beam elements.

- The accuracy and precision of the computation for various parameter choices, and also the performance of the algorithm in comparison to the time taken on a CPU are to be evaluated.
Jacobi solver on a GPU to solve Poisson equation

Spring 2011

Sarangarajan.V.Iyengar

Department of Mechanical Engineering
University of Wisconsin, Madison
Motivation

- Solving CFD problems are computationally expensive
- One of the important steps which takes the major share in computational time is solving the Poisson equation
- Since there is no analytic solution for the Navier-Stokes equation we opt for Iterative schemes.
- Some common techniques used are
  - Jacobi method
  - Gauss-Seidel method
  - Successive over relaxation method
  - Conjugate gradient method
Goal

- Midterm project:
  - Implementation of GPU based Jacobi solver.

- Final project:
  - Solving 2D Incompressible Laminar N.S. equations using the GPU based Jacobi solver.
ME964 Final Project: Monte Carlo Ray Tracing using CUDA

Zigfried Hampel-Arias
Final Project Proposal
14 April, 2011
**Introduction:**
Cosmic Rays produce $N \approx O(10^{10})$ particles in air
$N \sim$ energy $\rightarrow$ estimated from Cerenkov light in water tank

**Motivation for Work:**
Need faster tank sim (always...)
Monte Carlo

**Detector Function:**
Relativistic particles from air showers emit Cerenkov light in water tank
GPU Implementation

Simulation:
Photons are created independently and non-interacting
  -> Embarrassingly Parallel!!!
  -> Threads don’t share information
  -> 1 photon/thread

Method:
-Ray trace photons in tank....wait for it...
  - Reflections off tank walls NOT specular
  - Water absorption/scattering non-deterministic
  - Detection by Photomultiplier Tube stochastic

.... Distributions, distributions, distributions....
**GPU Implementation**

**Proposed Solution:**
- Photon’s life has three states (device functions):
  - Reflection
  - Absorption (dead)
  - Detection

- Best to group photons in similar life-states
- Readout detector response after all photon’s die
- DO PHYSICS!!!
Background

- Magneto-Rheological (MR) Fluid is a fluid with magnetic particles suspended in it
  - When a magnetic field is applied, properties of the fluid change (i.e. viscosity)
- Common application today: Magneride shocks

Modeling aspects

- Can treat each magnetic particle as a sphere
- Each particle is acted on by a magnetic force, hydrodynamic force, and a wall force
- This becomes an N-Body simulation
Current Techniques used to reduce computation time

- Taking advantage of the fact that $F_{ij} = -F_{ji}$
  - Enables interaction between each particle to be calculated only once
- Use of neighbor list
  - Determines which particles are “neighbors” to a particular particle
    - Indicates which particles might interact with a particular particle later in the simulation
  - Only iterate over the “neighbor” particles when calculating the force
    - Opposed to iterating over every particle to see which are within range
  - If calculated too frequently code slows down
    - Simply just iterating over every particle anyway
  - If calculated too infrequently, may not include particles that are interacting with a particular particle later in the simulation
Goals of this project

- Use knowledge of CUDA when calculating the forces between particles.
  - Starting with a cluster of particles, observe how they interact when including only magnetic forces

- Determine the speedup of the code when compared with sequential code.

- Time permitting, include periodic boundary condition, forces associated with the system walls, and hydrodynamic drag.
2D Navier-Stokes Solver

- Conservation Equations
  - Mass
  - X-momentum
  - Y-momentum

- Numerics
  - 2\textsuperscript{nd} order Central Difference derivatives
  - Adams-Bashforth time integration
Physical Setup

- Rectangular domain, grid
- Uniform inlet flow, advective outflow
- Shear-free walls parallel to flow
- Gaussian vortex initial condition
Parallel Numerics

- Embarrassingly Parallel
  - Initialization
  - Time integration
- Pressure-velocity coupling
  - Solve matrix using Poisson solver
  - Iterative methods can be parallelized
    - Not necessarily deterministic